

Paul M. Clarke, Peter Elger, and Rory V. O'Connor. 2016. Technology enabled continuous software development. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery* (CSED '16). ACM, New York, NY, USA, 48-48. DOI: <http://dx.doi.org/10.1145/2896941.2896943>

Technology Enabled Continuous Software Development

Paul M. Clarke

School of Computing

Dublin City University, Ireland

Lero - Irish Software Research Centre

+353-1-700-7021

Paul.M.Clarke@dcu.ie

Peter Elger

NearForm Ltd.

Suite 420 Mountain View

CA 94040, USA

+1-916-235-6459

Peter.Elger@nearform.com

Rory V. O'Connor

School of Computing

Dublin City University, Ireland

Lero - Irish Software Research Centre

+353-1-700-5643

Rory.OConnor@dcu.ie

ABSTRACT

Given that organizations need to innovate and release software in faster parallel cycles of days or even hours, there are good reasons why new practices are being adopted in industry. In this paper, we present the case of a highly responsive process that is driven by tooling technology and which facilitates continual delivery of software at up to hourly intervals. This approach can inform academic and practitioner dialogue on current challenges and potential solutions, and on the evolution of new 'better' practices.

CCS Concepts

• Software development process management

Keywords

Software development lifecycle; Lean software development.

1. INDUSTRIAL PRACTICE – A CASE

Software development organizations are under more pressure than ever before to evolve software intensive systems through the release of valuable software in increasingly shorter time durations. At one time, software releases would occur one or two times per year, however with current competitive market opportunities this has been reduced to weekly, daily and hourly time periods.

One software organization at the forefront of such innovative process design is NearForm Ltd., a software development company with a presence in the US and Europe and which continues to witness impressive growth through the continual delivery of high quality software to some of the largest companies in the world, including blue chip financial institutions. *Value* is a key focus in the NearForm lifecycle and it is concerned with an acute responsiveness to client needs (be they new features or defect resolutions). The organization works to a regular 5 day iteration for software development, deploying working software weekly through a standard feature bundle. For customers who desire even higher levels of responsiveness, the company operates continual deliveries of working software, at sometimes daily and even hourly intervals. While regular iterations can be predictable from the outset, continual analysis of the value stream ensures that each iteration may be re-planned in real time, delivering the highest possible level of *value* from organizational capacity.

Whilst it is acknowledged that tooling can affect the design of a software process [1], the impact of technology on shaping the process in this case is extremely profound and initially appears to run contrary to the Agile Manifesto value of '*Individuals and interactions over processes and tools*'. Within NearForm the continual software evolution and delivery is made possible through the aggressive incorporation of contemporary and predominately open source software tools. While the speedy delivery of innovative features is a vital enabler of competitive

advantage, it is only effective if it is accompanied by reliable and high quality deployments. To this end, the availability of cutting edge technology and tools has driven the current form of the organizational software processes, with a strong focus on both deployment and application quality. There are four key technologies driving the process architecture: (1) Java-script and Node.js which enables extremely rapid code development by utilizing the same programming language across the entirety of the system; (2) alongside a distributed micro-services architecture, under which the system is broken down into a set of discreet co-operating processes, typically each service is of the order of several hundred lines of code only. (3) This architectural approach is coupled with a continuous deployment model, layered over the Docker container engine, whereby individual services (or several services at a time) may be deployed without perturbing the system as a whole. (4) Finally the company ensure quality through steps such as code commit hooks via GitHub (for distributed revision control and source code management) and the Travis CI tool set. Together, these technologies allow the company to perform well under a time and materials contract basis, whereby clients are initially attracted through the rapid delivery of a prototype in 10 days, and thereafter, regular iterations of new working software are reviewed every 5 days.

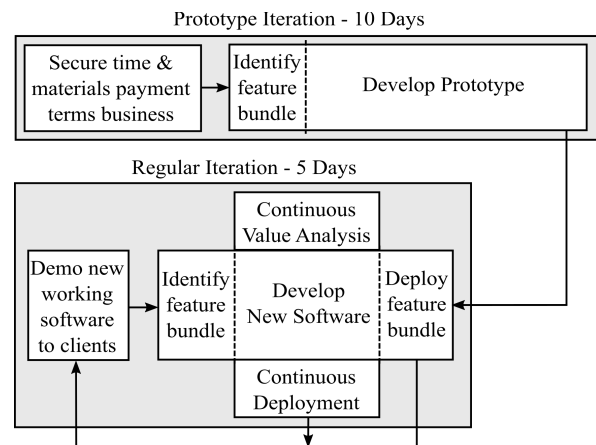


Figure 1. Nearform Technology Driven Process Lifecycle.

The ever-increasing demands on developers will continue to cause major differences in how processes are designed, deployed and evolved. Therefore it is essential that evolving practices and success stories such as the one described herein are documented and exchanged.

2. REFERENCES

- [1] Clarke, P., O'Connor, R. V. 2012. The situational factors that affect the software development process: Towards a

Paul M. Clarke, Peter Elger, and Rory V. O'Connor. 2016. Technology enabled continuous software development. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery* (CSED '16). ACM, New York, NY, USA, 48-48. DOI: <http://dx.doi.org/10.1145/2896941.2896943>

comprehensive reference framework. *Information and*

Software Technology, 54(5): 433-447.